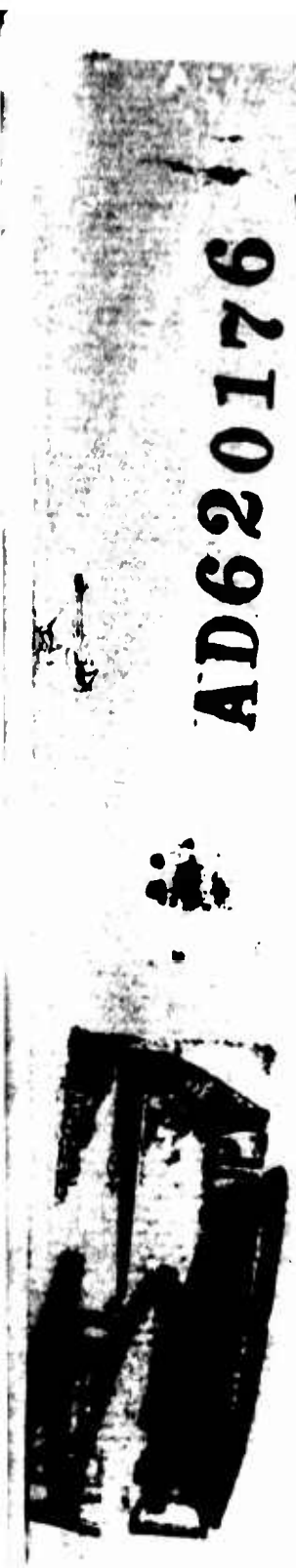
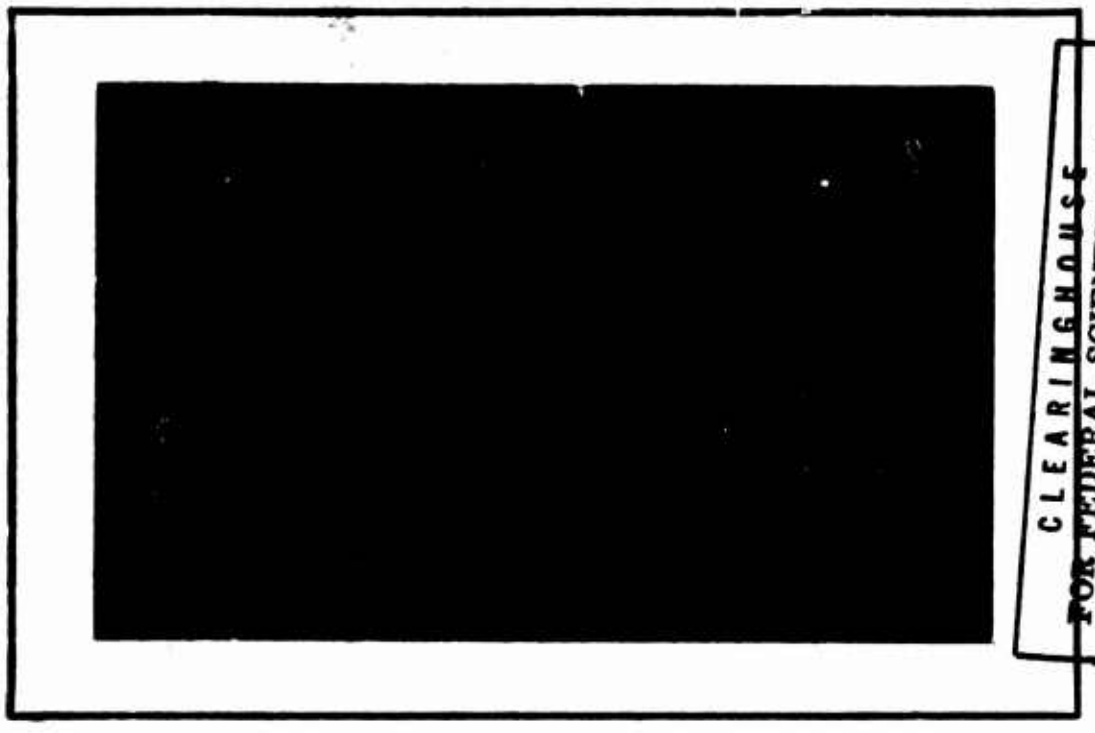
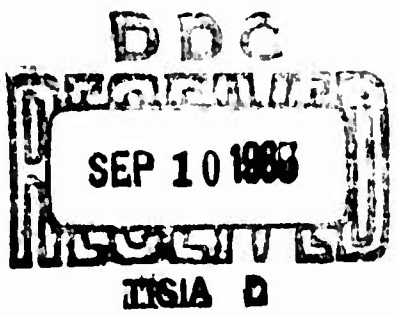
AD620176





CLEARINGHOUSE
FOR FEDERAL SCIENTIFIC AND
TECHNICAL INFORMATION
Hardcopy | Microfiche | 36 PP
$2.00 | $0.50



Carnegie Institute of Technology

Pittsburgh 13, Pennsylvania

GRADUATE SCHOOL of INDUSTRIAL ADMINISTRATION

William Larimer Mellon, Founder

Management Sciences Research Report No. 49

AN EXTENSION OF THE BOUND ESCALATION METHOD FOR INTEGER PROGRAMMING: A PSEUDO PRIMAL-DUAL ALGORITHM OF THE GOMORY ALL-INTEGER VARIETY

by

Fred Glover

July, 1965

MANAGEMENT SCIENCES RESEARCH GROUP
GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION
CARNEGIE INSTITUTE OF TECHNOLOGY
PITTSBURGH, PENNSYLVANIA 15213

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie Institute of Technology, (under Contract Nonr 760(24) NR 047-048 with the U. S. Office of Naval Research). Distribution of this document is unlimited. Reproduction of this paper in whole or in part is permitted for any purpose of the United States Government.

ABSTRACT

The bound escalation method is an algorithm for solving integer linear programs that is closely related to the all-integer integer programming algorithm developed by Ralph Gomory. In the bound escalation method the pivoting process is decomposed into two separate stages. In the first stage the problem matrix is subjected to a series of nonsingular integer linear transformations (corresponding to integral row additions and subtractions in the problem tableau) to create a new problem exhibiting a special structure called the bounding form. The second stage then operates on the bounding form to obtain lower bounds for a subset of the current problem variables, and this information is utilized in conjunction with the successively derived problem matrices to guarantee convergence to the optimal solution for the original problem in a finite number of steps.

The extension of the bound escalation method developed in this paper is still more closely allied to Gomory's all-integer algorithm, and indeed may alternately be regarded as an extension of that method. The extension arises out of a special case in which the bounding form consists of a single column vector. With the bounding form thus restricted, the Gomory all-integer algorithm may be regarded as an application of the bound escalation method to constraints that may be weaker than those in the tableau, the purpose of the restricted application being to obtain a single-column bounding form in one step. This interpretation leads to the consideration of "one-step" transformations that yield a bounding form without employing weakened constraints. Drawing on other considerations developed in the paper, the transformation we select results from pivoting on a Gomory inequality in violation of the

rule that maintains the tableau dual feasible. This choice is accompanied by a recovery phase in which dual feasibility is once again restored, allowing comparison with the primal method of R. D. Young. The rules governing the recovery guarantee a net advance toward the optimal solution, thus also invoking an analogy with the primal-dual algorithm of ordinary linear programming. Other variations also arising out of the results of the paper are briefly discussed in a concluding section.

1. Introduction.

In this paper an extension of the bound escalation method is presented that enables the customary dual feasibility requirement in the integer programming tableau to be systematically violated. The purpose of this extension is to enlarge the number of solution strategies at our disposal for solving integer programming problems. At this stage we do not know a great deal about matching algorithms to specific problems; however, the efficiency of the extension proposed here in solving some of the problems examined encourages the hope that it may find useful application as we gain increased knowledge of its particular strengths and weaknesses in relation to various problem types.

The general course of the presentation in this paper will be as follows. A description of the integer programming problem and the notation to be used are given in Section 2. Beginning with Section 3, and continuing through its several subsections, an informal outline of the bound escalation method is presented, highlighting the concept of the bounding form and the relation of the bound escalation method to Gomory's all-integer method. The implications of the similarities and contrasts between the two methods, established within the context of the single-constraint bounding form, provide the basis for the extension developed in the following section.

Section 4, and particularly Section 5, which together contain the basic results of this paper, follow a more formal pattern. Those interested chiefly in the results of these sections, and less concerned about the background material, may skip directly from Section 2 to Section 4 without sacrificing the ability to follow the accompanying theorems and proofs.

Example problems illustrating the method are solved in Section 6, where special variations implied by the results of the preceding sections are also outlined.

2. Notation and Description of the Problem.

The integer linear programming problem may be written

(1) Minimize $wb + b_o$

subject to $wA \geq c$, $w \geq 0$ and w integer,

where b_o is a scalar, b is an $m \times 1$ column vector, A is an $m \times n$ matrix, c is a $1 \times n$ row vector, and w is a $1 \times m$ row vector of variables. We assume that A has the form (A^o, I), where I is the $m \times m$ identity matrix, and that c correspondingly has the form $(c^o, 0)$, where 0 denotes the $1 \times m$ zero vector. Thus, $wA \geq c$ is composed of the constraining relations $wA^o \geq c^o$ and $wI \geq 0$, so that the nonnegativity conditions on w are included in the general matrix inequality $wA \geq c$.

We shall additionally assume that the components of b, A, and c are integers (although rational numbers are permissible), and that the augmented matrix $(-b, A)$ is lexicographically (lex) negative[1] by row, thus satisfying the conditions of dual feasibility.[2]

We alternately represent problem (1) by the following tableau

$-b$	A
b_o	c

1. A nonzero vector is said to be lexicographically (lex) negative when its first nonzero component is negative. A lex positive vector is similarly defined.
2. Dual feasibility is satisfied whenever $b \geq 0$, although for convenience we shall frequently refer to dual feasibility as synonymous with the more restrictive conditions imposed on $(-b, A)$ above. We choose to deal with $(-b, A)$ rather than $(b, -A)$ -- which must then be lex positive — to establish notational uniformity between the present paper and [3], where this convention was employed to permit certain relationships involving the constraints summarized by $wA \geq c$ to be glimpsed more readily.

We designate the - b column of the tableau (including b_0) as column 0, and the first column of A (including c_1) as column 1 of the tableau, and so forth. For convenience in referring to the rows of the tableau, we designate the ith row by a^i,

$$a^i = (a_{i0}, a_{i1}, a_{i2}, \dots, a_{in}), \qquad i = 1, 2, \dots, m,$$

where $a_{i0} = -b_i$ and a_{ij} denotes the ijth element of the A matrix for $i, j \geq 1$. Similarly, we designate the bottom row of the tableau by a^0,

$$a^0 = (a_{00}, a_{01}, a_{02}, \dots, a_{on}),$$

where $a_{00} = b_0$ and $a_{0j} = c_j$ for $j \geq 1$.

3. An Outline of the Bound Escalation Method.

3.1. Optimality and strategic objectives.--We observe that if the c vector happens to be nonpositive, an optimal solution to problem (1) is immediately given by $w = 0$. For then $wA \geq c$ is satisfied, and also $wb + b_0 = b_0$, there being no smaller value possible for $wb + b_0$ when b is nonnegative and $w \geq 0$. Thus our objective in solving problem (1) will be to obtain in its place a new problem satisfying the following four conditions: (i) an optimal integer solution to the original problem is immediately determined by specifying an optimal solution to the new problem, (ii) the (-b, A) matrix for the new problem is lex negative by row, (iii) the c vector for the new problem is nonpositive, and (iv) the variables of the new problem are constrained to be nonnegative integers under the assumption that the original variables are so constrained.

As we have seen, the last three conditions assure that an optimal solution for the new problem will be obtained by setting all variables equal to zero. Condition (i) then assures that the original problem is thereby solved.

The foregoing of course corresponds to one of the usual strategies for solving linear programming problems, and the conditions (ii) and (iii) are customarily referred to as the conditions of dual and primal feasibility, respectively.

To create the new problem of the desired form, the bound escalation method operates on the tableau of the original problem to obtain a succession of new problems each satisfying conditions (i), (ii), and (iv), until eventually a problem is created that also satisfies condition (iii). Thus the tableaus for the successive problems may be represented in the same form as that of the original,[1] and the final tableau is obtained when the c vector is driven nonpositive. The solution to the original problem is then given by $w = -\hat{c}$, where $\hat{c}$ denotes the last m components of the final c vector (the portion that began as the 0 vector).

As we have outlined it to this point, the bound escalation method follows the same general pattern as the simplex method of linear programming. However, the bound escalation method permits only integer transformations to be applied to the tableau, in order to assure that each of the new variables so created may be expressed as an integer linear combination of the original variables plus an integer constant.[2]

Each integer transformation corresponds to adding integer multiples of a row a^s to the other rows. These row operations, appropriately restricted so as to maintain the current problem variables nonnegative and the upper tableau lex negative, are first applied to the (-b, A) matrix to create the structure called the bounding form. Thereupon, positive lower bounds are

---

1. It is not necessary, however, that the successive A matrices be of the form (A^o, I).
2. The resemblance here to Gomory's all-integer method is apparent, and will be elaborated upon shortly.

determined for a subset of the problem variables, and the bottom row a^0 is modified in such a manner as to advance the problem toward solution. Thus, specifically, the bound escalation method consists of two alternating stages that may be summarized as follows:

1. Stage 1: Use row operations that preserve lex negativity in the (-b, A) matrix to create a new problem in nonnegative integer variables exhibiting a structure called the bounding form.

2. Stage 2: Operate on the bounding form to obtain positive integer lower bounds for a subset of the problem variables (the lower bounds for the remaining variables of course being zero). Replace the c vector by the vector $c - \hat{w}A$, and replace b_o by $b_o + \hat{w}b$, where $\hat{w}$ denotes the vector of the indicated lower bounds.

3. Repeat the foregoing process until the current c vector becomes nonpositive, whereupon an optimal solution to the original problem is given by setting w equal to the negative of the last m components of the current c vector (or a^0).[1]

3.2. The bounding form.—We now consider the structure that provides the cornerstone for the method. In applying the first stage of the bound escalation method we desire to create a tableau which upon suitable indexing may be partitioned as follows.

- b	D	*
	Q	
b_o	d	*

1. Following the criterion established by Gomory, to insure that the c vector will eventually be driven nonpositive it suffices to use any rule for creating a bounding form that will eventually include a column r in the bounding form structure for which c_r has not otherwise been driven nonpositive.

Here D is a p x p square matrix with positive entries along the main diagonal and nonpositive entries everywhere else. Q is a matrix composed entirely of nonpositive entries, and d is a p component row vector at least one of whose entries is positive.[1] Under these conditions we refer to the special structure of D, Q, and d as constituting the bounding form. We are unconcerned with the portions of the tableau marked with an asterisk.

When $d \geq 0$ ($d \neq 0$), the assumption that the problem has a feasible solution implies that the inverse of D, D^{-1}, exists and consists entirely of nonnegative components. In addition it may be shown that the nonnegative vector $D^{-1}d$ gives lower bounds for the first p components of the current w (not necessarily the original w). Any fractional components of $D^{-1}d$ may of course be rounded upward to the next highest integer to give integer lower bounds for the elements of w.

It is also possible to obtain lower bounds in a similar fashion when d contains negative as well as positive components, though such considerations will not be pursued here.

Beginning with the specified lower bounds, the bound escalation method continues by an elementary process of incrementing these bounds as necessary to obtain a resulting vector of lower bounds d*, $d^* \geq 0$, such that $d - d^*D \leq 0$. The vector d* may then be used to give the first p components of $\hat{w}$, thus enabling the current c vector to be modified as described in Section 3.1.

For the purposes of the paper we shall consider only the case in which D, Q, and d constitute a single column of the tableau. In this instance, d and D each are composed of a single positive element, d_o and d_1 respectively, so that the constraint associated with this column may be written

---

1. D, Q, and d need not appear explicitly in the tableau, but may be generated by any nonnegative linear combination of the tableau columns (excluding column 0).

$$(2) \qquad d_1 w_1 + \sum_{i=2}^{m} d_i w_i \geq d_o,$$

where $d_i \leq 0$ for $i \geq 2$. The integer lower bound $\hat{w}_1$ for w_1 in this case is simply $< d_o/d_1 >$, [1] as may be inferred immediately from the fact that $\sum_{i=2}^{m} d_i w_i \leq 0$, and hence $w_1 \geq d_o/d_1$.

The special form of the constraint (2), which gives the simplest instance of the bounding form structure, also supplies a condition under which row operations in the tableau preserve nonnegativity in the resulting variables. Thus, if multiples of row a^1 are added to the other rows, thereby replacing the original w_1 variable by a new w_1, we can be assured that the new w_1 is nonnegative if the tableau exhibits a constraint such as (2). Of course, this conclusion also holds when $d_o \leq 0$ provided $< d_o/d_1 > \geq 0$ $(d_o/d_1 > -1)$.

Generally speaking, as long as the tableau is kept lex negative, negative multiples of a^1 ("row subtractions") are always permissible since the constraint $w_1 \geq 0$ (implicitly if not explicitly associated with the tableau) is thereby changed into the desired form. On the other hand, positive multiples ("row additions") of a^1 are less frequently available since they cannot be justified by reference to a transformed nonnegativity constraint. In spite of this, row additions are generally desirable whenever they are possible, since the new variables so defined are usually smaller than those they replace. This may be seen from the fact that adding a^1 to a^2, for example, corresponds to replacing w_1 by the variable $w_1' = w_1 - w_2$. The contrary conclusion of course holds concerning row subtractions.

---

1. We use the notation $< x >$ to denote the least integer greater than or equal to x.

3.3. Creating a single-constraint bounding form.—We now consider how to manufacture a constraint of the form of (2) from at least one column of the tableau. We assume that a^o is not all nonpositive (disregarding a_{oo}) and that the problem has a feasible solution. The first step is then to select a column r ($r \geq 1$) of the tableau for which a_{or} ($= c_r$) > 0. If only one[1] of the components of that column is positive (other than a_{or}) then the associated constraint

$$\sum_{i=1}^{m} a_{ir} w_i \geq a_{or}$$

is already in the specified form. Otherwise, we may create a constraint such as (2) by the following general rule.[2]

(i) Select any two positive components a_{pr} and a_{qr} of column r ($p, q \geq 1$). By choice of indexing, assume a^p is lex larger than a^q.

(ii) Replace a^q by $a^q - a^p$, designating the resulting row as the new a^q.

(iii) Repeat the process until only one positive a_{ir} remains for $i \geq 1$.

At each application of the foregoing process the tableau is maintained lex negative and one of the positive coefficients in column r is decreased by subtracting one of the other positive coefficients from it. Thus, it is clear that eventually all but one of the positive a_{ir} will become nonpositive.[3]

3.4. Relation to the Gomory inequalities.-- While many variations are subsumed under the foregoing general rule, a variation of particular

---

1. At least one a_{ir} ($i \geq 1$) must be positive if the problem has a feasible solution.
2. M. L. Balinski [1] has pointed out that this rule establishes a close correspondence between the first stage of the bound escalation method and the euclidean algorithm. Of course, in order to maintain the new variables non-negative, keep the tableau lex negative, and create a bounding form, we employ specializations not ordinarily required of the euclidean algorithm.
3. This is assured by the assumption that the components of A are integer (or rational).

interest leads to the creation of a constraint such as (2) in a single step. To see how this variation arises, suppose, for example, that a^1 is the lex largest row in the upper tableau having a positive component in column r. If a_{1r} is the largest of the positive a_{ir} for $i \geq 1$, then a^1 may be subtracted once from all a^i such that $a_{ir} > 0$ $(i \geq 2)$ to give a constraint in the form of (2) at once. But if a_{1r} is smaller than some of the other a_{ir}, we may work instead with a weaker constraint in which a_{1r} is increased by any desired amount. Clearly, it may not be necessary to make a_{1r} the largest positive coefficient in the weakened constraint, since possibly more than a unit multiple of a^1 may be subtracted from the other rows.

It turns out that, when a_{1r} is increased only as much as necessary, and all permissible row additions are made after the required subtractions, the same transformation of the tableau results as by the usual pivoting procedure of Gomory's all-integer algorithm.[1]

However, it is not always desirable to try to create a constraint such as (2) in a single step, particularly if a substantially weaker constraint than the original must be employed in order to do so. Therefore, different methods have been proposed for creating a single-constraint bounding form than the one prescribed by the all-integer algorithm. One such method, suggested in [3], provides the starting point for the extension developed in this paper.

3.5. Background of the proposed extension.—The method to which we have reference consists simply of carrying out all row operations with the row that would ordinarily be used for pivoting with the simplex method. Specifically, the largest possible multiple of the

---

1. The complete correspondence follows upon altering the a^0 row by determining the lower bound for the resulting w_1. More generally, selecting the size of a_{1r} corresponds to determining the parameter λ in the Gomory inequality

$$\sum \langle a_{ir}/\lambda \rangle w_i \geq \langle a_{or}/\lambda \rangle.$$

selected row is subtracted from each of the other rows, subject to the two qualifications that the upper tableau must be maintained lex negative and no multiple is used which is larger than that required to make the a_{ir} of the other rows nonpositive. Until a bounding form is created, the lex smallest row such that $a_{ir} > 0$ must of course be excluded in determining the row for carrying out row subtractions. Thereafter, all row additions that preserve the bounding form structure are carried out, and the a^0 row is adjusted in the fashion indicated earlier.

While this rule can be shown to be more effective than the approach of the all-integer method in certain cases, it is also frequently less effective, primarily due to the fact that the lex negativity requirement may severely restrict the row operations available.

In this paper we present a method that uses the *same row* choice as the preceding rule, but which overcomes some of the limitations of this rule by allowing certain of the rows of the (-b, A) matrix to become lex positive. In fact, the method begins precisely as the one outlined above except that the lex ordering of the tableau is ignored, thus making it possible to obtain a bounding form on each step without weakening any of the problem constraints. As our previous discussion indicates, the resulting transformation of the tableau corresponds to one obtained from pivoting on an appropriate Gomory inequality, although this inequality will be prescribed by the all-integer method only if recourse to a weakened constraint is unnecessary to assure lex negativity in the upper tableau.

When one or more of the tableau rows becomes lex positive, the phase of restoring the (-b, A) matrix to lex negativity is immediately iniated. At the conclusion of this phase the a^0 row is lex larger than at any previous point when (-b, A) was lex negative.

Thus, in brief, our extension of the bound escalation method may be regarded as a "pseudo" primal-dual method of the Gomory variety, exhibiting certain features in common both with Gomory's all-integer algorithm and with R. D. Young's primal method.[1] We now turn to specifying the form of this extension precisely.

4. The Extension.

To facilitate the ensuing exposition we introduce the following additional notation and definitions. Relative to a specified column r, define

$$a^{i}* = (a_{io}/a_{ir}, a_{i1}/a_{ir}, \ldots, a_{in}/a_{ir}),$$

for each i, $1 \leq i \leq m$, such that $a_{ir} \neq 0$. Further, let $a^{s}*$ denote the lex largest $a^{i}*$ such that a^{i} is lex negative and $a_{ir} > 0$. Finally, let $a^{v}*$ denote the lex smallest $a^{i}*$ such that a^{i} is lex positive and $a_{ir} < 0$.

Then, beginning with the tableau of Section 2, the integer programming method proposed in this section may be described as follows.

1. If $a_{oj} \leq 0$ for all $j \geq 1$, an optimal solution to the problem is given by $w = -\hat{a}^{o}$, where $\hat{a}^{o}$ denotes the last m components of a^{o}. Otherwise,

2. Select[2] a column r such that $a_{or} > 0$, $r \geq 1$. Identify the row a^{s}, where $a^{s}*$ is defined as above, and let

$$\bar{a}^{o} = a^{o},$$

$$\bar{a}^{i} = a^{i} - \langle a_{ir}/a_{sr} \rangle \bar{a}^{s} \text{ for } i \geq 0, \ i \neq s.$$

3. Update the tableau so that $a^{i} = \bar{a}^{i}$ for all $i \geq 0$. If

---

1. Another primal integer programming algorithm that employs Gomory inequalities is due to Ben-Israel and Charnes [2], although the Ben-Israel - Charnes method requires the solution of a series of intermediate auxiliary problems.

2. We assume that the rule of choice operates within the framework specified by Gomory in [4].

the resulting a^i are all lex negative for $i \geq 1$, return to 1.

Otherwise,

4. Identify the row a^v, where a^{v*} is defined as above, and let

$$\bar{a}^v = -a^v,$$

$$\bar{a}^i = a^i - \langle a_{ir}/\bar{a}_{vr} \rangle \bar{a}^v \text{ for } i \geq 0,\ i \neq v.$$

5. Return to 3.

A few preliminary observations are in order. Since $x + 1 > \langle x \rangle \geq x$ for any x, the definitions $\bar{a}_{ir} = a_{ir} - \langle a_{ir}/\bar{a}_{sr} \rangle \bar{a}_{sr}$ and $\bar{a}_{sr} = a_{sr} > 0$ together imply that $0 \geq \bar{a}_{ir} > -\bar{a}_{sr}$ $(i \neq s)$ after each application of Step 2. This means then that $\bar{a}_{sr}$ is the only positive $\bar{a}_{ir}$ for $i \geq 0$, and $\langle \bar{a}_{ir}/\bar{a}_{sr} \rangle = 0$ for all $i \neq s$. Thus, from the discussion of the constraint (2) in Section 3.2, the new w_s variable created by the transformation defined at Step 2 must be nonnegative, and of course integer. The same conclusions apply concerning Step 4, replacing s by v. The nonnegativity of the new w_s and w_v may alternately be justified by observing that the transformations are also given by pivoting on the Gomory inequality $\sum \langle a_{ir}/\lambda \rangle w_i \geq \langle a_{or}/\lambda \rangle$, where $\lambda = \bar{a}_{sr}$ at Step 2 and $\lambda = \bar{a}_{vr}$ at Step 4. The selected pivot rows, a^s and a^v respectively, may of course be different from those specified by the ordinary linear programming pivot rules applied with reference to the Gomory inequalities .

To prove that the method converges we have two principal objectives: first, to show that whenever some a^i $(i \geq 1)$ becomes lex positive when the tableau is updated following Step 2, the method will restore the upper tableau to lex negativity after a finite number of iterations of Step 4; and second, to show that the a^o vector will be lex larger on each visit to Step 1 than

on the previous visit. The Gomory proof of convergence for the all-integer algorithm then applies[1] for the appropriate choice of r at Step 2, and our goal will be achieved.

We may roughly outline the strategy of our proofs as follows. By the preceding remarks, each time Step 4 is visited, whether via the route from Step 2 or from Step 4 itself, exactly one component of column r will be positive. The row in which this component appears will be lex negative, hence this row corresponds to the current a^s. We will undertake to prove that two other important characteristics of the tableau will also be preserved at each visit to Step 4: (i) all a^i* for lex positive a^i ($i \geq 1$) will be lex larger than the current a^s*, and (ii) all lex positive a^i will be lex smaller than $-a^s$. The latter fact, dependent upon the former, implies that $-a^s$ must itself be lex decreasing, and hence that the number of iterations at Step 4 must be finite.

To prove also that a^o is lex larger at each visit to Step 1 than on the previous visit, we will in fact show that the amount of the lex increase in a^o must always be at least as large as that resulting from pivoting with the simplex method when a_{sr} is the selected pivot element.

5. Lemmas, Theorems, and Proofs.

In what follows we intend to identify a^1 with $\bar{a}^s$ or $\bar{a}^v$, as appropriate, and to identify a^2 with one of the other tableau rows. Hence we define $\bar{a}^2 = a^2 - \langle a_{2r}/a_{1r} \rangle a^1$, for some specified r, $0 \leq r \leq n$, such that $a_{1r} \neq 0$. (When referring specifically to the method it will of course be true that $r \geq 1$ and $a_{1r} > 0$.) Similarly, we define a^{i*} as in the preceding section for each i such that $a_{ir} \neq 0$.

---

1. This applicability is assured by the fact that $a_{or} \leq 0$ each time the tableau is updated at Step 3.

Theorem 1. Consider vectors a^1 and a^2 such that $a_{1r} \neq 0$, $a_{2r} > 0$. Then

(i) a^{1*} is lex larger than a^{2*}

if and only if:

(ii) $\bar{a}^2$ is lex negative

or

(iii) $\bar{a}_{2r} < 0$ and $\bar{a}^{2*}$ is lex larger than a^{1*}.

To simplify the proof of this theorem we state and prove the following three lemmas.

Lemma 1. Let a^1 and a^2 be given as in Theorem 1, and assume that $\bar{a}_{2r} = 0$. Then (i) is true if and only if (ii) is true.

Proof of Lemma 1. To prove this lemma we shall derive an expression for $\bar{a}_{2j}$ that will be used again in proving Lemma 2. By definition $\bar{a}_{2j} = a_{2j} - \langle a_{2r}/a_{1r} \rangle a_{1j}$. Also, $a_{2j}^* = a_{2j}/a_{2r}$ and $a_{1j}^* = a_{1j}/a_{1r}$, so that $a_{2j} = a_{2j}^* a_{2r}$ and $a_{1j} = a_{1j}^* a_{1r}$. Substituting for a_{2j} and a_{1j} in the definition of $\bar{a}_{2j}$ thus yields

$$\bar{a}_{2j} = a_{2j}^* a_{2r} - \langle a_{2r}/a_{1r} \rangle a_{1j}^* a_{1r}.$$

By the definition of $\bar{a}_{2r}$ we have

$$a_{1j}^* \bar{a}_{2r} = a_{1j}^* a_{2r} - \langle a_{2r}/a_{1r} \rangle a_{1j}^* a_{1r},$$

and thus from the preceding expansion of $\bar{a}_{2j}$ we obtain

$$\bar{a}_{2j} = a_{1j}^* \bar{a}_{2r} + a_{2r}(a_{2j}^* - a_{1j}^*). \tag{3}$$

When $\bar{a}_{2r} = 0$, the fact that $a_{2r} > 0$ implies by (3) that

(a) $\bar{a}_{2j} = 0$ if and only if $a_{2j}^* = a_{1j}^*$, and

(b) $\bar{a}_{2j} < 0$ if and only if $a_{2j}^* < a_{1j}^*$.

Let $\bar{a}_{2p}$ be the first nonzero component of $\bar{a}^2$. Then $\bar{a}^2$ is lex negative if and only if $\bar{a}_{2p} < 0$. Also, by (a) above, $a_{2j}^* = a_{1j}^*$ for $j < p$, and $a_{2p}^* \neq a_{1p}^*$. Therefore, a^{1*} is lex larger than a^{2*} if and only if $a_{2p}^* < a_{1p}^*$. But then by (b) we have that a^{1*} is lex larger than a^{2*} if and only if $\bar{a}^2$ is lex negative, which is what we desired to prove.

Remark 1: Lemma 1 hold by exactly the same proof when $\bar{a}^2$ is alternately defined to equal $a^2 - Ka^1$, where K is any number, thus increasing the generality of the preceding result. Lemmas 2 and 3 below likewise hold for this weakened definition of $\bar{a}^2$.

Lemma 2. Let a^1 and a^2 be any two vectors such that $a_{1r}, a_{2r} \neq 0$ and $\bar{a}_{2r} \neq 0$. Then for any j, $0 \leq j \leq n$,

$a_{1j}^* = a_{2j}^*$ if and only if $a_{1j}^* = \bar{a}_{2j}^*$.

Proof of Lemma 2. Under the assumption that $\bar{a}_{2r} \neq 0$, we have $\bar{a}_{2j}^* = \bar{a}_{2j}/\bar{a}_{2r}$ by definition. Substituting the value of $\bar{a}_{2j}$ given by equation (3) of Lemma 1 into this last equation gives

$$\bar{a}_{2j}^* = a_{1j}^* - a_{2r}(a_{2j}^* - a_{1j}^*)/\bar{a}_{2r}. \tag{4}$$

For a_{2r} and $\bar{a}_{2r}$ nonzero, it is clear from (4) that $\bar{a}_{2j}^* = a_{1j}^*$ implies $a_{2j}^* = a_{1j}^*$, and conversely that $a_{2j}^* = a_{1j}^*$ implies $\bar{a}_{2j}^* = a_{1j}^*$.

Lemma 3. Let a^1 and a^2 be any two vectors such that $a_{1r}, a_{2r} \neq 0$ and $\bar{a}_{2r} \neq 0$. Assume, moreover, that $a_{2r}/\bar{a}_{2r} < 0$. Then $\bar{a}^{2*}$ is lex larger (smaller) than a^{1*} if and only if a^{1*} is lex larger (smaller) than a^{2*}.

Proof of Lemma 3. Excluding the case $a^{1*} = a^{2*} = \bar{a}^{2*}$, which is irrelevant to the conclusion of the lemma, it is assured by Lemma 2 that there exists an index q such that $a_{2q}^* \neq a_{1q}^*$, $\bar{a}_{2q}^* \neq a_{1q}^*$, and $a_{1j}^* = a_{2j}^* = \bar{a}_{2j}^*$ for $j < q$. But from equation (4) of Lemma 2, $a_{2r}/\bar{a}_{2r} < 0$ implies that $a_{1q}^* < a_{2q}^*$ if and only if $a_{1q}^* > \bar{a}_{2q}^*$. This proves Lemma 3.

Proof of Theorem 1. By definition $\bar{a}_{2r} = a_{2r} - \langle a_{2r}/a_{1r} \rangle a_{1r}$. Since $a_{1r} > 0$ and $a_{2r}/a_{1r} \leq \langle a_{2r}/a_{1r} \rangle$, we therefore have $\bar{a}_{2r} \leq 0$. If $\bar{a}_{2r} = 0$, then Theorem 1 is true by Lemma 1, while if $\bar{a}_{2r} < 0$, then $a_{2r}/\bar{a}_{2r} < 0$, and Theorem 1 is true by Lemma 3.

Remark 2. By the foregoing proof we see that Theorem 1 is also true if (ii) is altered to read "$\bar{a}^2$ is lex negative and $\bar{a}_{2r} = 0$," thereby increasing the

generality of the "only if" part of the Theorem.

Theorem 2. Let a^1 and a^2 be given as in Theorem 1 and assume in addition that a^1 and a^2 are both lex negative. Then if (i) is true and $\bar{a}^2$ is lex positive, it follows that $\bar{a}^2$ is lex smaller than $-a^1$. Moreover, the first nonzero component of $-a^1$ is larger than the corresponding component of $\bar{a}^2$.

Proof of Theorem 2. Let $\bar{a}_{2p}$ be the first nonzero component of $\bar{a}^2$. If $a_{1j} \neq 0$ for some $j < p$, then the assertion is immediately true. Also, $a_{1j} = 0$ for all $j \leq p$ is impossible since then $\bar{a}_{2j} = a_{2j}$ for $j \leq p$ and thus a^2 is lex positive, contrary to assumption. Thus suppose that a_{1p} is also the first nonzero component of a^1 (where it is given that $\bar{a}_{2p}$ is the first nonzero component of $\bar{a}^2$). Since a^{1*} is lex larger than a^{2*}, we have $a_{1p}/a_{1r} \geq a_{2p}/a_{2r}$, hence $a_{2r}/a_{1r} \leq a_{2p}/a_{1p}$, since $a_{1p} < 0$ and $a_{2r} > 0$. It follows that $\langle a_{2r}/a_{1r} \rangle \leq \langle a_{2p}/a_{1p} \rangle$ and $\langle a_{2r}/a_{1r} \rangle < a_{2p}/a_{1p} + 1$. From the definition of $\bar{a}_{2p}$, $\bar{a}_{2p} = a_{2p} - \langle a_{2r}/a_{1r} \rangle a_{1p}$, we therefore have

$$\bar{a}_{2p} < a_{2p} - (a_{2p}/a_{1p} + 1)a_{1p},$$

and hence $\bar{a}_{2p} < -a_{1p}$. This completes the proof.

Theorem 3. Consider two vectors a^1 and a^2 such that

$a_{1r} \neq 0$, $a_{2r} < 0$, $\bar{a}^2 \neq 0$.

Then

(i) a^{2*} is lex larger than a^{1*}

if and only if:

(ii) $\bar{a}^2$ is lex negative

or

(iii) $\bar{a}_{2r} < 0$ and $\bar{a}^{2*}$ is lex larger than a^{1*}.

Remark 3: The conclusion of Lemma 1 continues to hold, following the same pattern of proof as above, when a^1, a^2 and condition (i) are alternately given as in Theorem 3. Likewise, the conclusion of Theorem 2 holds under the same modifications when a^1 is lex negative and a^2 is lex positive.

Remark 4: When the assumption $a_{2r}/\bar{a}_{2r} < 0$ is replaced by $a_{2r}/\bar{a}_{2r} > 0$ in Lemma 3, a similar argument yields the conclusion that $\bar{a}^{2*}$ is lex larger (smaller) than a^{1*} if and only if a^{2*} is lex larger (smaller) than a^{1*}.

Proof of Theorem 3. By reference to the preceding remarks, the proof of Theorem 3 mirrors that of Theorem 1.

Theorem 4. If any a^i, $i \geq 1$, is made lex positive as a result of the transformation defined at Step 2, then all a^i for $i \geq 1$ will become lex negative after a finite number of iterations of Step 4.

Proof of Theorem 4. Letting $\bar{a}^s$ correspond to a^1 and a^i to a^2 ($i \neq s$), it follows from Theorem 1 that if $\bar{a}^i$ is lex positive at Step 2, then $\bar{a}_{ir} < 0$. Hence, after updating the tableau, all lex positive a^i ($i \geq 1$) are candidates for a^v the first time Step 4 is visited. It also follows from Theorem 1 that a^{v*} is lex larger than the previous a^{0*}. Let it be imagined that Step 4 is broken into two parts, where first a^v is replaced by $\bar{a}^v$ (which we continue to refer to as $\bar{a}^v$ instead of the new a^v), but the remainder of the updating of the tableau is deferred, as before, until Step 3 is visited. Thus, by the above remarks, when a^v is replaced by $\bar{a}^v = -a^v$, but before any of the rest of the tableau is updated, then $\bar{a}^v$ satisfies the definition to qualify as the new a^0. Moreover, by definition, a^{i*} is lex greater than $\bar{a}^{v*}$ ($= a^{v*}$) for all lex positive a^i ($i \geq 1$, $i \neq v$). Thus, associating $\bar{a}^v$ with a^1 and associating the lex positive a^i with a^2 in Theorem 3, we see that each such a^i will remain lex positive after Step 4

only if $\bar{a}^{i}*$ is lex larger than $\bar{a}^{v}*$. On the other hand, any a^{i} other than the current a^{s} that is lex negative when the tableau is updated at Step 3 will remain lex negative after each subsequent iteration of Step 4. This follows from the fact that $a_{ir} \leq 0$ for all such a^{i}, and hence only a nonpositive multiple ($< a_{ir}/\bar{a}_{vr}>$) of the lex negative $\bar{a}^{v}$ will be subtracted from those rows due to the transformation specified at Step 4. Consequently, each subsequent iteration of Step 4, as well as the first, will insure that (a) $a_{ir} < 0$ for all lex positive a^{i}, (b) the current $\bar{a}^{v}$ corresponds by definition to the current a^{s} (when $\bar{a}^{v}$ replaces a^{v} but the rest of the tableau is not yet updated), (c) $a^{i}*$ is lex larger than $\bar{a}^{v}*$ for all currently lex positive a^{i} ($i \geq 1$), and (d) each lex negative a^{i} other than the one that was a^{s} before $\bar{a}^{v}$ replaced a^{v} will remain lex negative after the iteration of Step 4 is completed and the tableau is updated at Step 3. So long as these mutually interdependent conditions obtain we shall also have by Theorems 1, 2 and 3/and their accompanying remarks that all a^{i} that are lex positive at the end of Step 4 will be lex smaller than $-\bar{a}_{v}$, and hence that the new a^{v} determined at the start of the next iteration at Step 4 will be lex smaller than the previous a^{v}. Since the vectors are changed by integer amounts, the a^{v} vectors cannot be indefinitely decreased lexicographically in the indicated components and remain lex positive. Moreover, by Theorem 2 (and Remark 3) the first nonzero component of the old a^{v} must be larger than the corresponding component in the new a^{v}. Thus, eventually all a^{i} for $i \geq 1$ must be made lex negative. (The fact that the tableau rows are linearly independent assures that there will only be one vector qualifying for a^{v} at any given time, and hence that none of the a^{i} will be reduced to the zero vector.)

Remark 5. The last part of the foregoing proof may alternately be established by observing that $0 \geq a_{ir} > -a_{sr}$ on each visit to Step 4, and hence that the a_{ir} are steadily being increased (in integer amounts) toward 0 by the increasing value of $-a_{sr}$ (equal to the preceding $\bar{a}_{vr}$). But since condition

(a) of the foregoing proof must hold at each iteration, none of the a_{ir} for lex positive a^i can actually reach 0, and hence eventually all a^i must become lex negative.

Theorem 5. On each visit to Step 1, the a^0 vector will be lex larger than on the previous visit. Moreover, the amount of the increase will be at least as large as that resulting from pivoting on column r by the rules of the simplex method.

Remark 6. In proving Theorem 5 we will let a^0, $a^{\beta}*$, $\bar{a}^0$, and $\bar{a}^{\beta}$ denote the indicated vectors as they are defined when applying Step 2. In addition, for each iteration h ($h \geq 0$) at Step 4, let k_h denote the current $< a_{0r}/\bar{a}_{vr} >$, ρ_h denote the current $\bar{a}^v$ vector, ρ_h* denote the current $\bar{a}^v*$, and ρ_{hr} denote the current $\bar{a}_{vr}$. Thus, if $\bar{\bar{a}}^0$ denotes the final a^0 vector after the last iteration of Step 4, we may write

$$\bar{\bar{a}}^0 = \bar{a}^0 - \sum_h k_h \rho_h , \quad \text{and} \tag{5}$$

$$\bar{\bar{a}}_{or} = \bar{a}_{or} - \sum_h k_h \rho_{hr} . \tag{6}$$

Remark 7. By the definition of $a^{\beta}*$, a^{β} will be the pivot row when pivoting on column r with the simplex method. Also, the vector to replace a^0 by the simplex pivoting rules is $a^0 - a_{0r}a^{\beta}*$. We note that this results in a lex increase in a^0 since $a_{0r} > 0$ and $a^{\beta}*$ is lex negative.

Proof of Theorem 5. To prove the theorem we must show, by the foregoing remarks, that $\bar{\bar{a}}^0$ is lex greater than or equal to $a^0 - a_{0r}a^{\beta}*$. Since $\bar{a}_{0r} \leq 0$, from (6) we obtain

$$\bar{\bar{a}}_{0r} \leq \sum k_h \rho_{hr} . \tag{7}$$

By the definition of ρ_h* and ρ_{hr} we also have

$$\sum k_h \rho_h = \sum k_h \rho_{hr} \rho_h* . \tag{8}$$

As shown in the proof of Theorem 4, successive reapplications of Theorems 1 and 3 imply that ρ_h^* is lex larger than a^{s*} for each h. Since $k_h \leq 0$ and $\rho_{hr} > 0$, it follows from (8) that $\Sigma k_h \rho_h$ is lex less than or equal to $a^{s*} \Sigma k_h \rho_{hr}$. Also, since a^{o*} is lex negative, we have by (7) that $\Sigma k_h \rho_h$ is lex less than or equal to $\bar{a}_{or} a^{s*}$. Thus, from equation (5) we conclude that $\bar{a}^o$ is lex greater than or equal to $\bar{a}_o - \bar{a}_{or} a^{s*}$. Finally, using the fact that $\bar{a}_{sr} a^{s*} = \bar{a}^s$, the definitions of $\bar{a}_o$ and $\bar{a}_{or}$ yield $\bar{a}^o - \bar{a}_{or} a^{s*} = a^o - a_{or} a^{s*}$. This proves the theorem.

Theorem 6. The method specified in Section 4 will obtain an optimal solution in a finite number of steps for any problem that has a nonempty solution set and that satisfies the assumptions of Section 2.

Proof of Theorem 6. The proof is immediate from Gomory's convergence proof for the all-integer integer programming algorithm applied to the results of the foregoing theorems.

6. Example Problems and Comments.

We solve three example problems in this section to illustrate various aspects of the method of Section 4. The first problem presents a straightforward application. The second problem is included to illustrate how the problem may be solved before dual feasibility is restored at Step 4. The means for recognizing when a problem has been "prematurely" solved leads to the consideration of other methods based on the results of Section 5, which we also discuss. Finally, the third problem depicts a type of situation in which the method of Section 4 experiences substantial difficulty in restoring dual feasibility at Step 4. The implications of this behavior in terms of matching transformations to problem structure are considered briefly in the concluding remarks.

Example Problem 1.

$$\begin{aligned} \text{Minimize} \quad & 23w_1 + 17w_2 + 3w_3 + 7w_4 \\ \text{s.t.} \quad & 27w_1 + 20w_2 + 16w_3 + 17w_4 \geq 128 \\ & 22w_1 + 14w_2 - 9w_3 - 2w_4 \geq 45 \\ & w_1, w_2, w_3, w_4 \geq 0 \end{aligned}$$

Translating this into the tableau form of Section 2, we have

0.

→ -23	27	22	1	0	0	0
-17	20	14	0	1	0	0
- 3	16	-9	0	0	1	0
- 7	17	-2	0	0	0	1
0	128	45	0	0	0	0
	↑					

where we have inserted an additional partition between the A^0 and the I matrix.

By the procedure specified in Section 4 we observe, beginning with Step 1, that $a_{oj} > 0$ for $j = 1$ and $j = 2$, hence we proceed to Step 2. We select for column r at Step 2 the one that contains the fewest positive components.[1] Thus, in this instance, $r = 2$, as indicated by the arrow in the tableau above pointing to column 2. From the definition of a^{s*} we see that $s = 1$ when $r = 2$, hence the arrow pointing to row 1. The transformations prescribed at Step 2 yield at Step 3 the new tableau

1.

-23	27	22	1	0	0	0
→ 6	-7	-8	-1	1	0	0
-3	16	-9	0	0	1	0
-7	17	-2	0	0	0	1
69	17	-21	-3	0	0	0
		↑				

1. Our choice here is dictated by the conceptual framework underlying the bound escalation method. For more refined criteria, see [3].

In this tableau row a^2 has become lex positive. Therefore, with r still at 2, we proceed to Step 4 and apply the indicated transformation for v = 2. The updated tableau obtained at Step 3 is then[1]

2.

	-5	6	-2	-2	3	0	0
	-6	7	3	1	-1	0	0
→	-9	23	-1	1	-1	1	0
	-7	17	-2	0	0	0	1
	57	61	-5	-1	-2	0	0
		↑					

None of the a^i for $i \geq 1$ is lex positive, and we return to Step 1. Since $a_{01} = 61 > 0$, we proceed to Step 2 and let r = 1, this being the only choice. Now s = 3, and by applying Steps 2 and 3 we obtain

3.

	4	-17	-1	-3	4	-1	0
	3	-16	9	0	0	-1	0
	-9	23	-1	1	-1	1	0
→	2	-6	-1	-1	1	-1	1
	84	-8	-2	-4	1	-3	0
		↑					

Once again the upper tableau has lex positive rows. At Step 4 we determine that v = 4, thereby at Step 3 yielding the tableau

4.

0	-5	1	-1	2	1	-2
-1	-4	11	2	-2	1	-2
-1	-1	-5	-3	3	-3	4
-2	6	1	1	-1	1	-1
82	-2	-1	-3	0	-2	-1

1. This tableau may also be obtained after two pivots with Gomory's all-integer algorithm. Thereafter, however, the all-integer method requires 9 additional pivot steps before obtaining the solution.

All a_{0j} are nonpositive for $j \geq 1$, hence the problem is solved. From the last m components of a^0 the optimal solution is seen to be

$w_1 = 3,\ w_2 = 0,\ w_3 = 2,\ w_4 = 1.$

The next example problem to be solved contains only a single constraint.

Example Problem 2.

$$\begin{aligned} \text{Minimize} \quad & 3w_1 + 5w_2 + 9w_3 + 7w_4 + 13w_5 \\ \text{s.t.} \quad & 6w_1 + 15w_2 + 36w_3 + 23w_4 + 41w_5 \geq 398 \\ & w_1, w_2, w_3, w_4, w_5 \geq 0 \end{aligned}$$

For convenience we will not bother to write down the last five columns of the tableau corresponding to the I matrix and the zero vector, but will explicitly represent these columns only when they are changed from their original form. Thus, for the initial tableau we have

0.

-3	6
-5	15
→ -9	36
-7	23
-13	41
0	398
	↑

The arrows accompanying the tableau point to column r and row s. Since s = 3, the third column of the original I matrix will be modified by the transformation defined at Step 2. Thus, in the resulting tableau below this modified column is included following the modified columns 0 and 1, and w_3 is written above the new column to identify the variable with which the column is associated.

1.

		w_3
6	-30	-1
4	-21	-1
-9	36	1
2	-13	-1
5	-31	-2
103	-34	-12

At Step 4 the transformations are initiated to return the lex positive row to lex negativity. Since $v = 1$, the first column of the original I matrix will now be changed, yielding the column below w_1 in the tableau below.

2.

	w_3	w_1
-6	30	1	-1
4	-21	-1	0
3	-24	-1	2
2	-13	-1	0
-1	-1	-1	-1
102	-4	-11	-1

This tableau still contains lex positive a^i for $i \geq 1$, and Step 4 must therefore be repeated.

3.

	w_3	w_1	w_2
2	-12	-1	-1	2
-4	21	1	0	-1
-1	-3	0	2	-1
2	-13	-1	0	0
1	-1	-1	-1	0
102	-4	-11	-1	0

Once again repeating Step 4 we obtain

4.

		w_3	w_1	w_2
-2	12	1	1	-2
0	-3	-1	-2	3
-1	-3	0	2	-1
0	-1	0	1	-2
-1	-1	-1	-1	0
102	-4	-11	-1	0

The problem is now solved, and an optimal solution is given by

$w_1 = 1$, $w_2 = 0$, $w_3 = 11$, $w_4 = 0$, $w_5 = 0$.

For the preceding problem, we note that the optimal solution was already given in Tableau 2. Since some of the a_{io} for $i \geq 1$ were positive, however, the solution was not identified as optimal at that point. It would have been possible to make this identification, however, in the following way.

We shall create a new row from a^v ($v = 2$) in Tableau 2 by dividing a^v through by $-a_{vo}$ ($= -4$),[1] and then add this new row to the tableau by inserting it above the others. Since the variable associated with this row (call it z) must be zero in the final solution, we also adjoin the two constraints $-z \geq 0$ and $z \geq 0$ at the end of the tableau. Carrying out this procedure relative to Tableau 2, we obtain

---

1. If $a_{vo} = 0$, we instead divide through by the first nonzero component of a^v.

2A.

		w_3	w_1	$-z$	z
→	-1	$\frac{21}{4}$	$\frac{1}{4}$	0	-1	1
	-6	30	1	-1	0	0
	4	-21	-1	0	0	0
	3	-24	-1	2	0	0
	2	-13	-1	0	0	0
	-1	-1	-1	-1	0	0
	102	-4	-11	-1	0	0
		↑				

The new top row and the new columns are segregated by the added partitions. It is evident by its construction that the new row must qualify as $\bar{a}^v$ at Step 4.[1] Thus the method may be applied by proceeding from Tableau 2A instead of from Tableau 2.

It is unnecessary to carry out the computations in order to predict two things about the $\bar{a}^i$ that will be defined at Step 4. First, since the first component ($\bar{a}_{vo}$) of the new row is -1 and the components of column 0 are integers,[2] we may predict that $\bar{a}_{io} = 0$ for all lex positive $\bar{a}^i$ ($i \neq 0$). This follows from the results of the preceding section, which assure that all lex positive $\bar{a}^i$ will be lex smaller than $-\bar{a}^v$, and that the first nonzero component of $-\bar{a}^v$ will be larger than the corresponding component of the lex positive $\bar{a}^i$.

The second thing to be observed is that, in the present case, $\langle a_{or}/\bar{a}_{vr} \rangle = 0$ (for $a_{or} = -4$ and $\bar{a}_{vr} = 21/4$), and hence $\bar{a}^o = a^o$. This fact and the one just established assure that the feasible solution given by the bottom row of Tableau 2 (and 2A) must also be optimal. In short, we have established that $\langle a_{or}/\bar{a}_{vr} \rangle = 0$ is a sufficient condition for

---

1. Conceptually, we may imagine that the negative of the new row was adjoined at Step 3, whereupon this row would correspond by definition to a^v at Step 4. More generally, of course, we may adjoin any lex negative row a^h (to qualify as the new $\bar{a}^v$) such that a^{h*} is lex smaller than the current a^{v*} and lex larger than the current a^{o*}.

2. By permitting rational numbers in the tableau, it suffices more generally to select the first component $\bar{a}_{vo}$ of the adjoined row to be $-1/k$, where ka_{ij} is an integer for all i and j. If column 0 already consists of nonpositive components, then $\bar{a}_{vo} = 0$, and our remarks have reference instead to the first column j such that $\bar{a}_{vj} \neq 0$. To demonstrate that a feasible solution is optimal, however, consideration may be limited as above to column 0.

a feasible solution to be optimal, where $\bar{a}_{vr}$ is determined as outlined above (but not by reference to the unexpanded tableau).

If adjoined rows and columns are actually employed in solving the problem, and not simply as a means of checking for optimality, then it will eventually be possible to restore the tableau to its original size.[1] This approach of adjoining rows may also be used at Step 2 to prevent the a^1 from becoming lex positive in the first place. There are clearly a number of possible variations, and by following the appropriate rules the tableau need not be expanded to the extent depicted by our illustration each time a new variable is added. To insure convergence it is of course necessary to have some means for assuring that the succession of rows and columns added to the tableau will not be unending.

For our last example, we now turn to a very simple problem that illustrates a situation in which the method of Section 4 encounters serious difficulty in re-establishing dual feasibility.

Example Problem 3.

$$\begin{aligned} \text{Minimize} \quad & 1w_1 + 28w_2 \\ \text{s.t.} \quad & 1w_1 + 45w_2 \geq 98 \\ & w_1, w_2 \geq 0 \end{aligned}$$

0,

	-1	1	1	
→	-28	45	0	
	0	98	0	0
		↑		

The next three tableaus are written without additional comment.

---

1. By selecting one or the other of the adjoined columns (which will always be the negative of each other) as column r, and persisting in this, if necessary, after their bottom row components are 0, eventually there will remain only one row of the tableau with nonzero components in these columns, at which time the indicated row and columns may be dropped. The optimal solution may of course be obtained before this size reduction process is completed.

1.

27	-44	1	-1
-28	45	0	1
84	-37	0	-3

2.

-27	44	-1	1
26	-43	2	-1
84	-37	0	-3

3.

25	-42	3	-1
-26	43	-2	1
84	-37	0	-3

One may infer from the structure of this problem that after six more steps we will obtain

19	-36	9	-1
-20	37	-8	1
64	0	-8	-2

This tableau gives an optimal solution by the remarks relating to the previous example problem. However, to restore dual feasibility by the method of Section 4 we may project by inference that 20 additional steps are required, at which point we obtain

-1	18	-27	1
0	-17	28	-1
64	0	-8	-2

Two interacting features of the tableau bequeathed by Step 2 appear to have contributed to the difficulty encountered at Step 4: (1) a^{1}_{4} and

a^{2_4} were nearly the same, and (ii) the components a_{3_0} and a_{3_r} of the vector $a^3 = a^1 + a^2$ were small in absolute value relative to the corresponding components of both a^1 and a^2.

The extent to which these features may apply more generally to characterize situations in which the transformations of Section 4 should be bypassed and others employed in their place is not yet known. However, a number of related considerations are evidently involved in determining what types of transformations should be employed — e.g., the frequency with which structures that are difficult for the method arise in practice, the ability to predict the result of several iterations of Step 4 for these difficult structures, and the availability of criteria for restoring dual feasibility at an earlier point than otherwise permitted by uninterrupted application of Step 4. We do not at present know a significant amount about these considerations, but can only acknowledge their relevance in determining the uses to which the results of the preceding sections may be put.

REFERENCES

[1] Balinski, M. L., "Integer Programming: Methods, Uses, Computation," Mathematica, 1965.

[2] Ben-Israel, A. and A. Charnes, "On Some Problems of Diophantine Programming," Cahiers du Centre d'Etudes de Recherche Operationnelle, Vol. 4, Brussels, 1962.

[3] Glover, Fred, "A Bound Escalation Method for the Solution of Integer Linear Programs," Cahiers du Centre d'Etudes de Recherche Operationnelle, Vol. 6, Brussels, 1964.

[4] Gomory, Ralph E., "All-Integer Integer Programming Algorithm," in J. F. Muth and G. L. Thompson, eds., Industrial Scheduling, Prentice Hall, 1963.

[5] Young, R. D., "A Primal Integer Programming Algorithm," Ph.D. Thesis, Graduate School of Industrial Administration, Carnegie Tech, 1964.